論文　**Paper**

統合知的CASEシステムにおける エキスパートシステムの系統的構築法

陳　　慧*・河野善彌**

(2002年12月20日受付，2003年1月20日改訂)

A Systematic Construction Method of Expert Systems for an Integrated Intelligent CASE System

Hui Chen* and Zenya Koono**

Synopsis: This paper introduces a method of constructing expert systems for automatic software design in an integrated environment. This integrated environment may be applicable from top-level system architecture design, data flow diagram design down to flow chart design and coding. The system is consists of these CASE tools and respective expert systems with automatic design capability by reusing past design. The construction way of these expert systems is based on systematic acquisition of design knowledge stemmed from a systematic design work process of well-matured developers. The design knowledge is automatically acquired from respective documents and stored in the respective knowledge bases. By reusing it, a similar software system may be designed automatically. In order to develop these expert systems in a short period, these design knowledge is expressed by the unified frame structure and, functions of the expert system units are partitioned to mono-functions and then standardized components. As a result, the design cost of an expert system can be reduced to standard work procedures. Another feature of this paper is to introduce the integrated environment for automatic software design. This system features an essentially zero start-up cost for automatic design resulting in substantial saving of design man-hours in the design life cycle, and the expected increase in software productivity after enough design experiences are accumulated.

1. はじめに

ソフトウェアの膨張はとめどもなく続き，常に自動化が求められている[11]。ソフトウェアの自動設計には，知識の問題等の大きな技術課題が多くあり，作業のみを自動化しても全ての負担が解消する訳ではない。筆者らは自動化システムを求めて，人の設計に倣うエキスパートシステムによるソフトウェアの設計自動化を研究している。これは設計の最下流（例えば構造化チャート）から上流（例えばデータフロー図，機能構造図）に遡り，部分毎の自動化をボトムアップに実現し，徐々にエキスパートシステム構築技術を高度化して自動化率を高める長期構想に基づいている。

筆者等の方針は初めから完全な自動設計システムを目指すのではなく，CASEツールとエキスパートシステムの組合わせ省力効果を高める事を可能にする方法である。最初開発された

* 国士舘大学21世紀アジア学部兼情報科学センター
** 東亜大学大学院

知的 CASE (Intelligent Computer Aided Software Engineering: ICASE) システム[3]は，下流設計の構造化チャートを用いて設計図面から設計知識を自動的に獲得し，それらを知識ベースに蓄える。獲得した知識を再利用することによってプログラミングフェーズを自動化する。簡単化した方式ではあるが，経験を重ねると自動化率は高まる。この成功に基づき，設計の最下流から上流に遡って，上流設計の機能構造図及びデータフロー図へ拡大し，統合的な知的 CASE ツール[4]が開発された。これにより，上流設計からコーディングまで適用できる標準化された設計プロセスを自動化することができる。これらの知的システムは商用描画ツールとそれぞれのエキスパートシステムから構成される。従来エキスパートシステムの開発にはかなりの時間がかかり，その期間の多くはエキスパートシステムの構築に費されている。開発の容易化，短期化および品質の向上のため，システムの構築を標準化することが必要である。本論文はこれらのため，設計知識を統一的に表現し，エキスパートシステムの構築を標準化する方法を報告する。

本研究は，高い成熟度のソフトウェア開発組織を対象エキスパートとし，エキスパートの知識体系の基本モデルとして階層的な工程をとる。設計の中心は自然言語による階層展開の連鎖になり，階層展開の概念の親子関係をそのプロセスの単位的な知識とする。設計図面から設計知識を系統的に獲得し，ボトムアップにそれを系統的にエキスパートシステムを再構築し，自動設計を行わせる。設計情報，設計知識の構成方法を定め，標準化する。また各機能を分割し，標準化し，母体エキスパートシステムを構築し，各種の設計知識をオープンエンドに追加して自動化する機能を与える。これにより上流レベルのシステム設計からデータフロー図，構造化チャートとコーディングまでの自動化システムの開発ができ，かつ容易化する。

この論文は，2 章で設計知識モデルの原理部分を説明し，3 章で獲得した知識モデルに基づき，エキスパートシステムを系統的に構築する方法を説明する。統合的な知的 CASE システムの構成について 4 章で述べる。5 章はエキスパートシステムを構築する省力化達成の度合いを評価し，人の知的作業の習熟効果を説明し，省力効果ができる事を示す。6 章はまとめである。

2. ソフトウェアの設計工程と知識体系

本研究は，高成熟度組織を対象エキスパート，その特徴とする階層的な設計工程を知識モデルにとる[10]。その標準的な設計工程は階層的な設計工程であり，これを階層的な設計文書体系が区切っている。SE (Systems Engineer) が外部仕様を担当し，標準機能を組み合わせて顧客のシステムの仕様書を設計する。製品の構成は標準化されており，その構成に合わせた技術別の階層的組織が各設計作業を分担する。各構成員は，繰返し同種作業を経験するから，

作業に早く習熟し，組織としても作業方法が標準化されていく。これは高い生産性を実現する鍵になっている。設計文書体系は階層的で豊富であり，文書の記述レベルは規定により統制され，工程間のインターフェースを明確に規定する作用を果たす。これは工程毎の設計結果のチェックに役立ち，高い品質を達成する鍵になっている。

系統的に設計知識を獲得し，それにより設計を再現して自動設計できる。設計情報は階層的に展開する。単位的な設計での入出力関係である展開関係を「設計ルール」と名付け，設計知識として図面から採取する。自然言語概念の展開であるとき，それは固定的関係であり，再利用可能である。この設計知識は，小さな進行段階毎に正確に作った図面から，誰でも，確実に，高い再現性と高い信頼度で，そして系統的に獲得できる。上位階層では，下位工程を順次実行させる状態遷移状の設計知識が顕著である。これに関連して作業終了や設計結果の評価などの完了判定の知識，次の作業の方向づけ等の設計知識がある。設計工程を調べると，これらの機能単位毎の設計知識モデルが得られる。逐次近似的に階層的工程を繰返して細分化して行けば，これらの設計知識は幾らでも詳しくモデル化できる。これらによりそれぞれの設計知識を厳密かつ正確に獲得して体系的に再現できる[2]。

3.　エキスパートシステム構築の方法

2章で述べた体系的な知識は，エキスパートシステムに系統的に構成できる。この方法は，エキスパートシステムのモデルとして当初の階層的工程の枠組みをとり，個々の単位的な個別の工程毎に小さなエキスパートシステム(以下ユニットという)はソフトウェア工学的に作る。簡単な下位工程からエキスパートユニットを作り，ボトムアップに枠組にはめ込んで行けば，幾らでも人の設計に近付ける事ができる。作図の設計知識である「設計ルール」を順次つなぎ合わせると階層的な設計知識の木になるから，これを知識ベースに蓄える。着目する図面上での入力に対して推論エンジンは知識ベースを検索して展開関係を図面上に再現させる。当初の入力を与えると，当初の最終図面と同じ出力が得られる。設計知識が足りる場合は図面を自動設計でき，不足の場合には当該部分を人が設計し，その新しい知識を追加する。これは設計経験を重ねるにつれより多くの設計知識を蓄え，図面設計の自動化率が高くなる方式であって，設計者が経験を増すにつれて作業が高速化するのと同様な特性を示す。この戦略に従えば，如何ようにでも，オープンエンドに，どこまでも系統的に設計を自動化し高度化する事が原理的に可能になる。このような系統的な構成をとれば，エキスパートシステムの系統的な構築の実体は，個々のエキスパートユニットの系統的な構築に帰着する。

3.1 設計知識

本システムは主な3つの展開プロセスの文書，機能構造図（Function Structure Diagram, FSD [5]），データフロー図（Data Flow Diagram, DFD）及び構造化チャート（Problem Analysis Diagram, PAD [7]）を設計対象とする。図1はこれらの設計図面での展開の例と図のシンボルを統一的に表すフレームの構成を示す。これらの設計知識の定義は以下に述べる。

(1) 機能構造図FSDの設計ルール

図1の中央の最上位に機能構造図の展開例を示し，機能モジュールMは次の機能モジュールM1～M3を階層的に展開する。ここで，シンボルMは親とし，シンボルM1～M3は子供とし，シンボルMからM1～M3への階層的な展開は親と子の関係で表す。このような親子関係はFSD設計ルールと名づける。

(2) データフロー図DFDの設計ルール

図1の中央に一対のデータフロー図を示し，そこで，菱形はデータを示し，正方形は機能を示す。親としての単位DFD（D1-F1-D2シンボル群）は，子供としてもう一つのDFD（D11-F11-D12-F12-D13シンボル群）に展開される。このような親と子の関係は，DFD設計ルールと名づける。また親の入出力データと子供の入出力データが一致する設

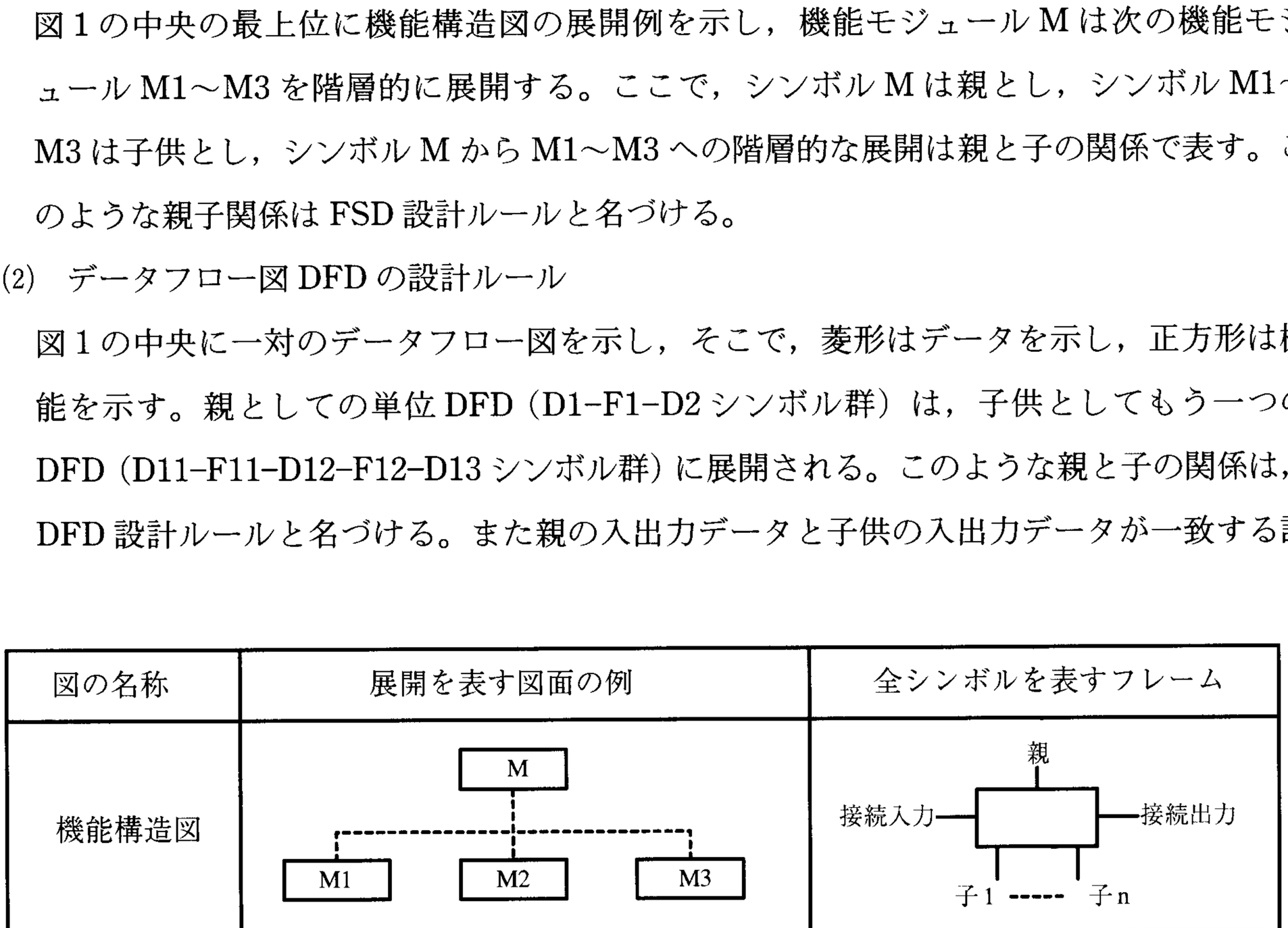

図1　統一的な図形表現形式

計ルールである。

(3)　構造化チャート図 PAD の設計ルール

図１の中央の最下位に構造化チャート図 PAD を示す。PAD は方形箱の処理，楔状箱の分岐，１本縦線つき方形箱の反復，２本縦線つき方形箱の呼び出しなど各種シンボルから成り立つ。この親 F2 シンボルから，いくつかの子供シンボル（F21…F24）に階層的に展開され，このような左から右へ展開関係を，PAD 設計ルールと名づける。ある設計段階の入力から出力への階層展開過程は１枚の図中の PAD 展開図形で表れるが，これは単位設計知識の連鎖からなる設計知識である。

設計ルールを自動獲得するために各図面の親子の関係を破線で対応を示す。設計システムで自動設計させるには上述各種の設計情報，設計知識のすべてを機械が理解できるように表現することが必要である。

3.2　設計情報の表記形式の統一

全システムを通じて設計情報を小さな段階毎に階層化し，標準化し，統一する。詳細設計情報の単位要素情報を図面のシンボルにとり，このシンボルの一般形を表す統一的な表記方法を求める。ここではシステム機能図，データフロー図，構造化チャート PAD を対象として標準化する。統一図形の表現形式の規則は以下の通りである。

- 図面のシンボル及びそれに伴う自然言語表記を記すとともに展開の関係をも表現するため，設計情報を統一的なフレームで表し，設計ルールのための展開関係は親と子の関係で表現する。
- ある設計階程での流れに従って，シンボル間の関係を「接続入力」と「接続出力」で表す。あるシンボルへの入力を「接続入力」で，あるシンボルから出力を「接続出力」で表す。図１の右列のように図面毎の流れに沿って接続入出力シンボルをつなぎ合わせて図面を構成する。
- シンボルの種類とその中に記される自然言語による表記をシンボルの名称として各フレームに表記する。

以上により，これらの図面の全シンボルを全て同一形式に表現する。図２は設計ルールのフレーム表現の１例である。図の左部分は設計ルールのグラフの表現であり，入力から出力への階層的な展開を表す。図の右部分は設計ルールのフレームの構造を示し，各シンボルの表現を統一したため，統一された単位シンボルのデータ構造もこのように簡単にできる。システムの中に全ての設計情報が同じフレームで表現し，標準化する。そうすると設計情報の追加や変

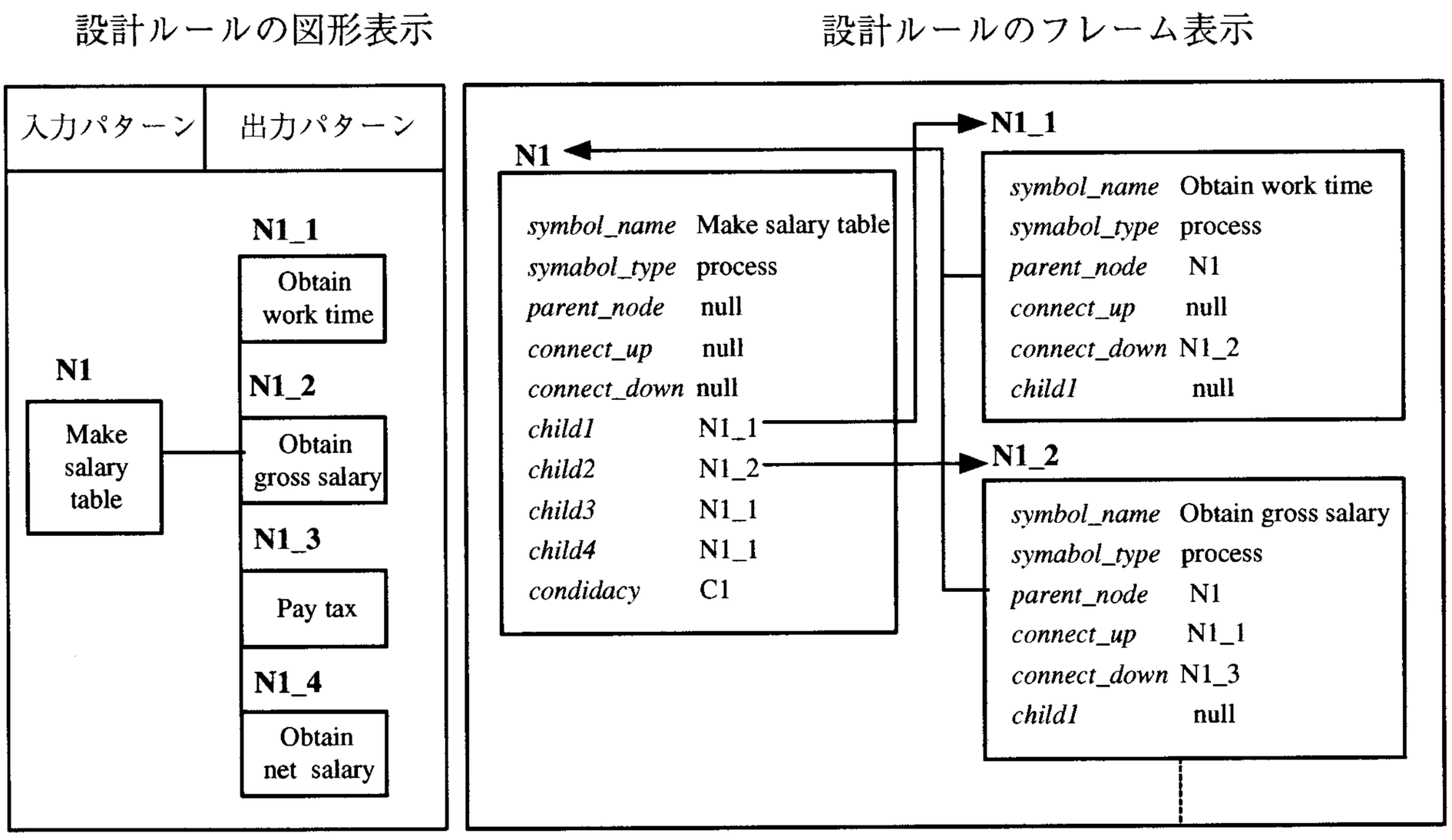

図 2　設計ルールのフレーム表現の 1 例

更などの各処理も簡単にできる。

3.3　設計知識の獲得原理

全て設計情報が統一されたため，各種の設計仕様書（例えば，FSD, DFD, PAD 設計図）から設計知識の自動獲得の原理は同じである。ここで，構造化チャート PAD を対象として設計知識の自動獲得の原理を図 3 により説明する。図 3 の一番下に PAD 設計図を示し，左端の一つの親シンボル *parent*（設計入力パターン）から出発して図形情報をたどり，それを右に階層展開した子のシンボル群 *child 1, child 2,* …,（設計出力パターン）を経由して出発点に戻る。これは一つの設計ルールにあたるから，図をたどりながらシンボル毎の図形情報と設計情報を取り出す。取り出した情報を解析すれば親と子の設計ルールが得られ，知識ベースに格納される。このような操作を繰り返せば全図面の設計ルールを自動的に獲得できる。

設計情報は階層展開になるので，知識関係は順次におおきくなり，ツリー状の知識ベースになる。設計が進むにつれてツリーを下りながら，新たな知識を知識ベースにどんどん加えていく。その最後の段階では，自然言語で表記した概念をソースコードに変換した結果を蓄積する。

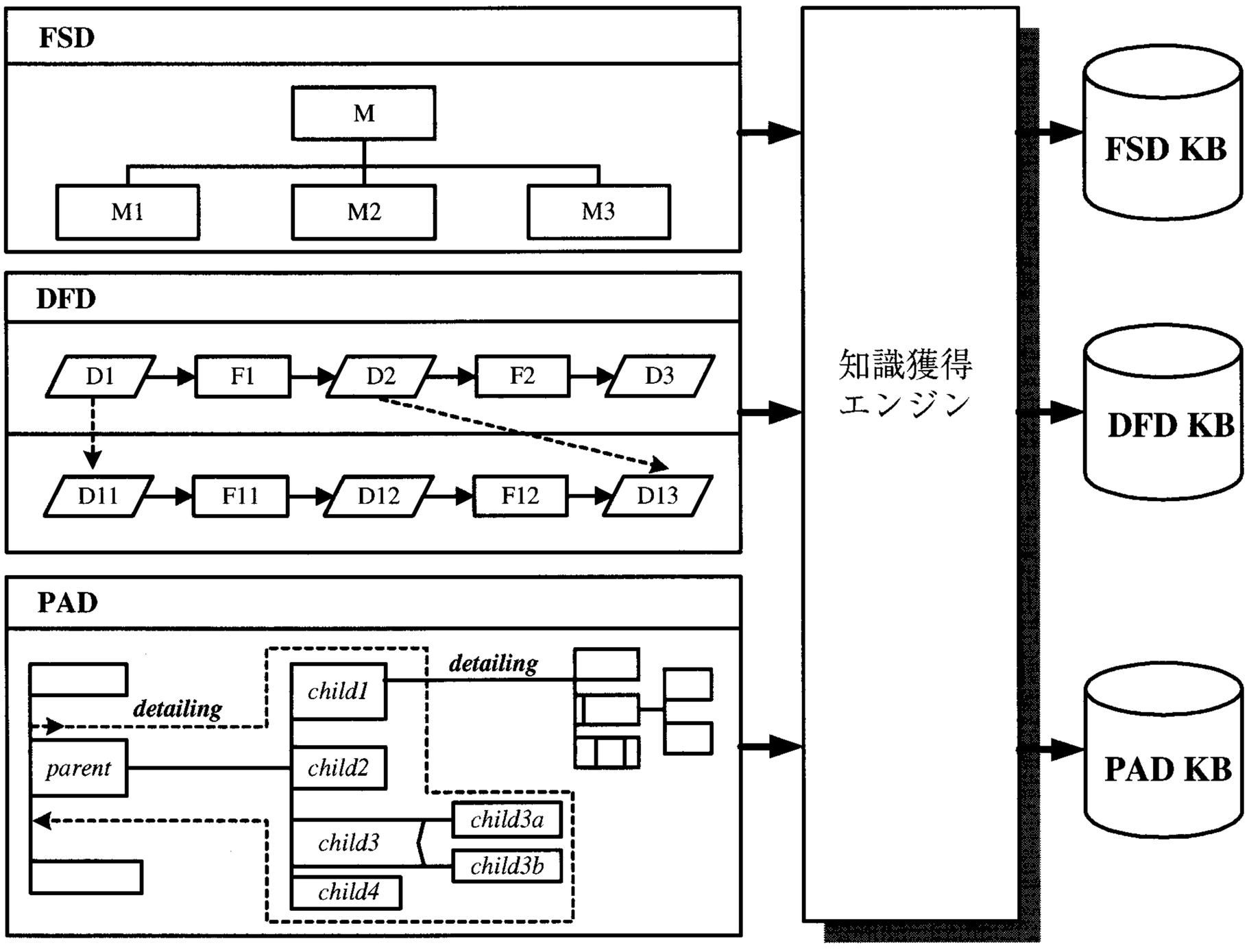

図 3　設計知識の獲得のユニット

3.4　知識ベースの構成

本システムの知識ベースはドメイン毎に分かれており，各ドメインは用語やプログラム条件の揃う範囲に選ぶ。ソフトウェアの構成方法によるが，設計対象の階層的なプログラム構成に階層的な知識ベースの構成が対応することになる。ここで設計ルールを再利用する技能レベルの知識ベースについて述べる。ドメイン毎の設計知識ベースは，図 4 に示すように 3 種類のデータテーブル「図面情報」,「シンボルデータ」及び「シンボル対応関係」から構成される。シンボル毎のシンボルの種別情報，シンボルの中に記入された表記，図面上の位置情報，候補数，ルール情報のポインター，図面情報のポインターなどの情報は「シンボルデータ」テーブルに保存される。自動設計時，検索された設計ルールを図面情報に高速に再現するため，各シンボルに関する図面情報は「図面情報」テーブルに保存される。シンボルの展開関係，すなわちシンボル毎の幾つかの候補，設計ルールは「シンボル対応関係」テーブルに保存される。

知識ベースの中で最も重要なテーブルは「シンボル対応関係」テーブルである。ここではシンボルの展開関係，即ち設計ルールは木構造を用いて記述される。1 シンボルはいくつかの設計ルールの候補を持つことがある。これは設計ルールと同様な展開関係として扱う。1 候補は

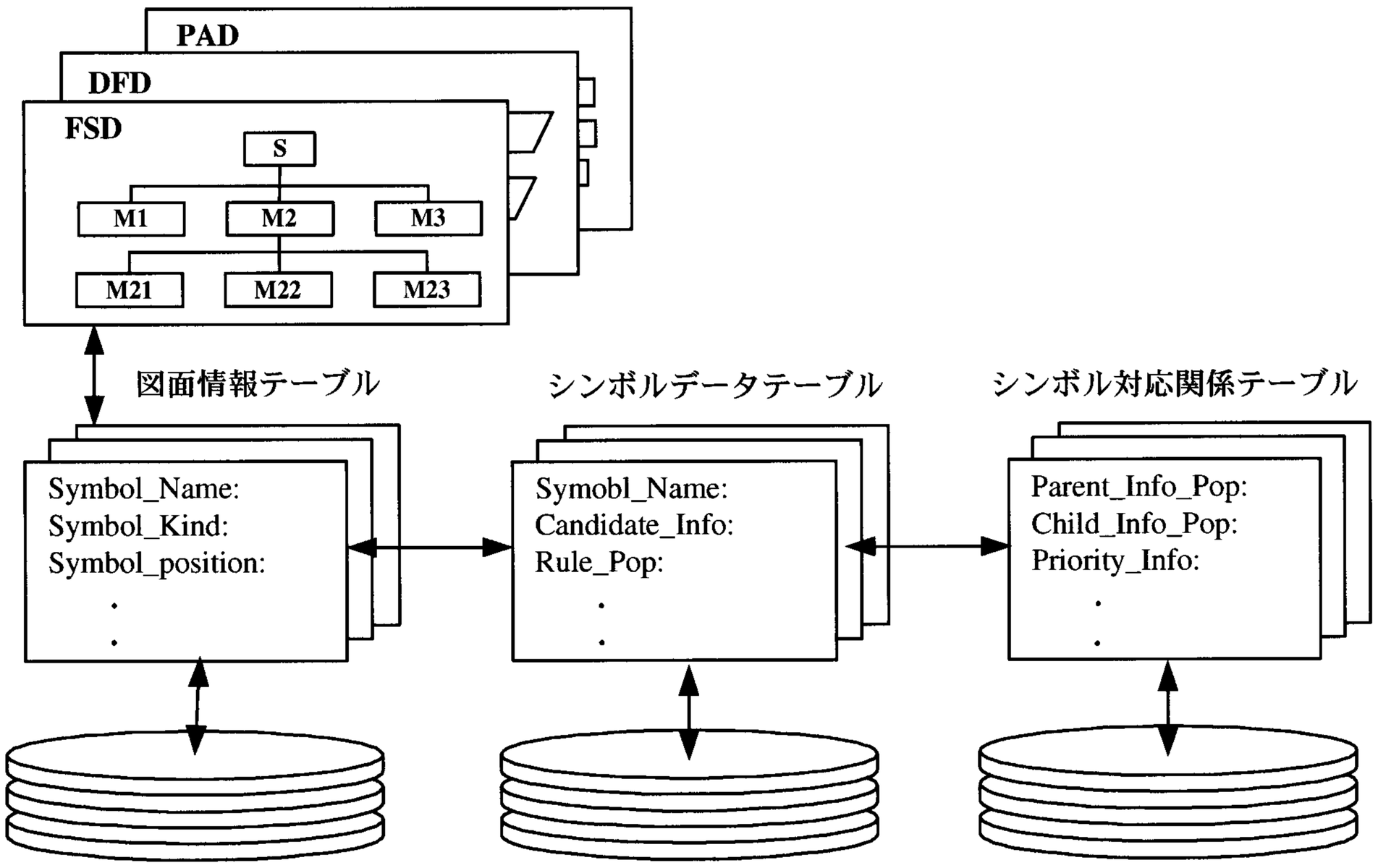

図 4　知識ベースの構成

いくつかのシンボルからなる設計ルールに展開される。このような各シンボルの間の関係情報は「シンボル対応関係」テーブルに格納される。また，DFD 設計知識から自動的に PAD 設計知識へ展開するために DFD と PAD の関係の設計情報も「シンボル対応関係」に追加されている。このようにすることにより，多くの情報を記録することができ，情報項目の拡大も容易にできる。設計では，書いたり消したり，修正することが多くある。設計情報（図面）を修正変更したら，知識情報の一貫性のため，設計知識ベースをも修正変更する。各種の検索も容易に行なえる。

3.5　各エキスパートユニットの標準化

このシステムでは階層的な文書体系を反映した階層的なデータ構造を持つから，ジャクソン法[9]が使える。これは階層構成を持つ入力から同じく階層構成を持つ出力への写像関係として部分機能を定める方法で，全体は階層的機能構成になる。このように構成を定めると，以後は部分機能レベルの詳細な設計しか自由度が残らない。これらについてユニットは以下に示す手順により更に小単位毎に細分化して，独立的な単機能化をはかり，機能を標準化する。これらを組合せ使用し，開発の容易化を達成する。

●着目する機能をその実現の為のデータフローに沿って最大抽象点で分断して単機能要素に

分解する。(Myers の S-T-S 分割[12])

- 単機能構成要素には単機能概念の機能名称を与えて独立モジュールとし，繰返し使用を可能にする。(全体を最小限の独立単位で構成するから，論理的な複雑度は必ず減少する。)
- 機能の具体化は入出力データの階層構成に歩調を合わせる。

以上を繰り返し行なう。

全段階の設計展開は，単一機能の入力から下位のいくつかの機能(群)への展開過程になり，データフロー図はアルゴリズムを示し，フローチャートは制御の流れを表す。このように具体化し，詳細化を繰り返すと，全レベルの展開はデータフロー図とフローチャートの対の積重ねになる。各所で単機能化することを繰り返すと，構成する要素は少数の基本的な要素のみになる。その結果はほとんどの最小の単位機能を関数に対応する。そこで，単位機能とソースコードを対応させ，標準部品を作る。これらの標準部品（関数）を再利用することにより最終展開結果をコードに容易かつ正確に変換できる。この方法によりわずか新しい関数を追加することで，以前に構築した構造化チャート PAD のエキスパートユニットの元に機能構造図及びデータフロー図のエキスパートユニットを簡単に構築することができた。この評価は第 5 章で述べる。

4.　統合知的 CASE システム

幅広く各種のプログラムに対応できる汎用的なシステムを目指して統合知的 CASE ツールシステムの母体開発の目的を達成するため，次の方針を立てる。

- 手軽に使用できて便利であり，設計効率向上に役立つ。
- 各種の CASE ツールと容易に統合できる。
- 機能拡張・変更・保守が容易に行なえる。

そこで，これらを達成するために実現上の具体的方針は下記のとおりとした。

- 既存 CASE ツールに知的機能を付加する。
- ユーザに対する親和性を重視する。
- 実行速度を向上させる。
- 各機能を相互に独立化する。
- 各モジュールを単機能化する。
- 各モジュールは単位的な要素であるから，横通しで見ると共通使用でき，時系列で見ると先行設計結果を再利用でき，これらで高い生産性と高い品質を達成できる。

以上述べた方針に従った統合知的 CASE ツールシステムを開発し，このツールを用いて設計する 1 例の画面を図 5 に示す。統合知的 CASE システムは CASE ツール群，設計エンジン

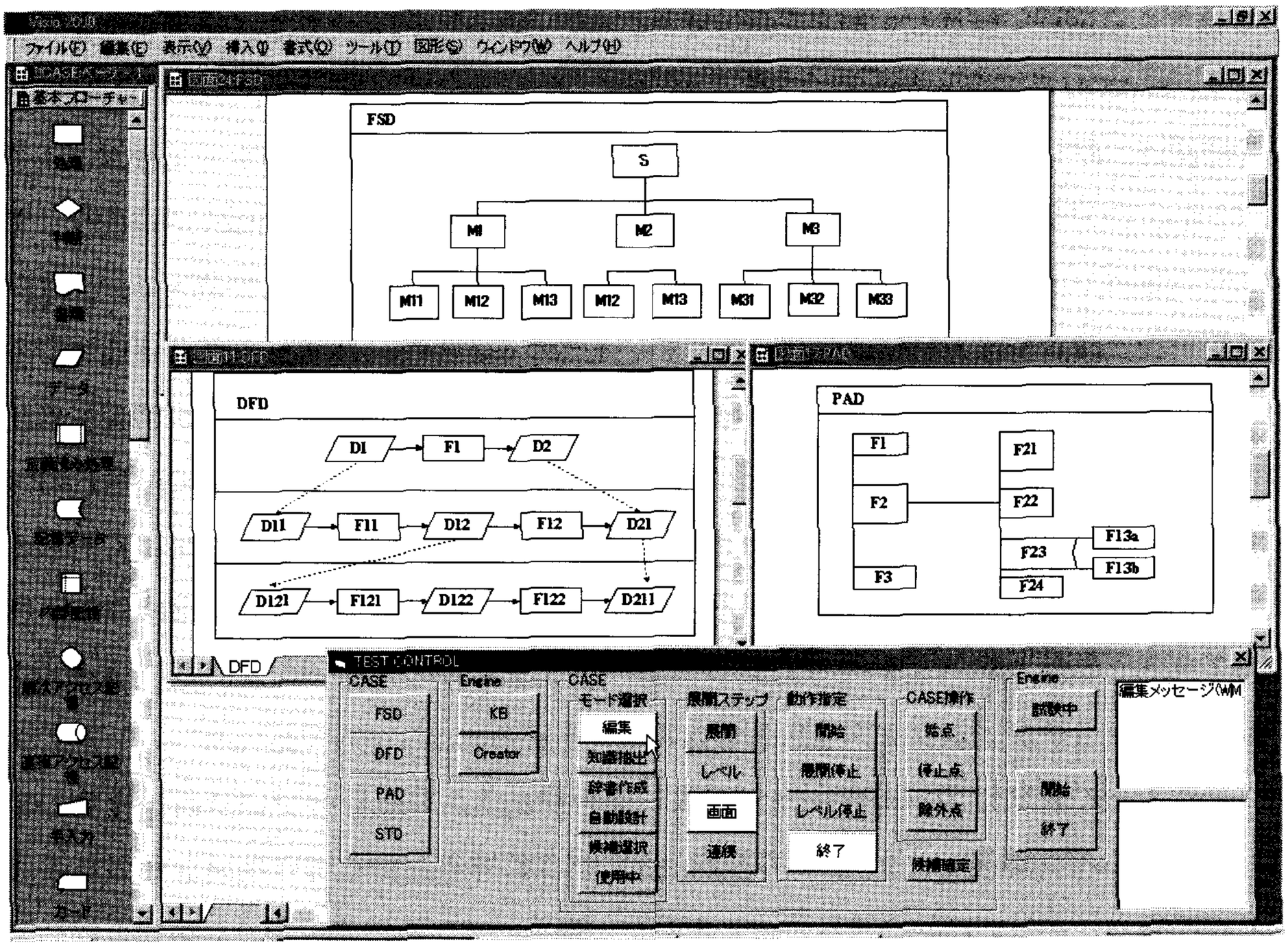

図 5　統合知的 CASE ツールの画面

群，知識ベース群，統合部及び制御部から構成される。

CASE ツール群はマイクロソフト社の商用 Visio [13]絵描きツールを利用し，FSDCASE ツール，DFDCASE ツール，PADCASE ツールと入出力情報の階層構造を書く構造辞書 (STD) CASE ツールがある。設計エンジン群は設計ルールの再利用エンジン及び設計知識を創り出すエンジン（設計ルールの自動生成やマクロデザインルールの自動生成）から構成される。統合部は各 CASE ツールと各エンジンの設計情報の授受や統合などの機能を果たす。制御部は使用者とのユーザインタフェイスや制御用メッセージをシステムの全体の制御を行う。各エンジンを CASE ツールと統合し容易に実現すること，またそれ以後の機能拡張・変更・保守を容易に行なえるようにするため，システムを相互に独立した機能で実現し，各機能を拡張有限状態機械（Extended Finite State Machine, EFSM）で構成する。また使用言語は Visual Basic 及び C であり，制御部，CASE ツール群とエンジンとの間のインターフェースは Windows APL (Application Program Interface) を利用する。ここで，設計ルールの再利用エンジンの機能について簡単に述べる。

1. 設計：設計者は各 CASE ツールの作図機能を用いてシステム，サブシステム，詳細論理など論理レベル段階的にシステムの設計図面を作成し，修正する。設計は全て階層的展開

であり，全階程で同様な手続きを繰り返し，最後はモジュール，関数，各単位機能に至るまで同一方法で詳細化できる。

2. 知識獲得：設計者は小さな段階毎の設計をおわるたびに，知識抽出指令をエキスパートシステムに送り，知識獲得エンジンを働かせ，ある設計の前後の展開関係を単位的な「設計ルール」として自動的に獲得する。獲得した設計知識を知識ベースに蓄積する。
3. 自動設計：設計情報と自動設計要求指令をエキスパートシステムに送って，推論エンジンを働かせる。設計情報により既存知識ベースから設計知識が検索され，設計情報に対応する設計ルールにより展開した情報が送り出され，CASE ツールの画面に表示される。この動作は次々と続く。設計ルールには複数の候補を持つ場合がある。設計者はこれらのうち最も要求に合うものを選ぶ。CASE ツールの編集機能を用い，設計結果の修正や拡張もできる。人または自動的な設計によりコード化を終ると，CASE ツールのコード変換機能によりプログラムソースリストを自動生成させる。

一つの親に複数の設計ルールが対応する場合がある。自動設計システムが複数の設計ルールから使用頻度の高いもの（第 1 候補）を自動的に選択して自動設計してある。これが不適当，あるいは他の設計ルールを見たい場合には，設計者は親設計情報と候補表示要求指令をエキスパートシステムに送り，推論エンジンを働かせる。この自動設計は単純に設計ルールをつなぎ合わせるから，人が候補選択をする必要が生じる。また，単純につなぎ合わせた結果が巨視的に目的に適合しない場合もありえる。そこで，適当な進行毎に個々の自動展開結果の整合を確認する。なお，設計ルールの自動生成及び複数候補の選択を自動生成するエンジン[1]も開発したが，ここでは省略する。

5. 評　　価

5.1 知識ベースの規模特性

今回の新しい知識ベースは従来の知識ベース（PAD のみ）にインターフェースの変更に伴

表 1　旧知識ベースと新知識ベースの規模比較

項目＼対象	旧 KB (PAD のみ)	新 KB (FSD, DFD, PAD)	内訳	
			旧 KB から再利用	新規追加
総ソース行数*	768	1332	895	437
モジュール数	22	28	22	6

* 総ソース行数：データ及び関数の定義はヘッダファイルに入れるため，総ソース行数に除外．コメントも除外．

う改造を施し，さらに FSD/DFD/PAD 図面を扱えるように拡大し，また DFD より自動的に PAD へ生成することができるようにした。新旧知識ベースの規模比較を表 1 に示す。

今回の知識ベースの機能は旧知識ベースの機能より何倍も拡大したが，規模はただ約1.74倍の大きさになった。表 1 はその内訳である。旧知識ベースから再利用したモジュールでは127行増加している。これは多種図面（FAD, DFD, PAD）を扱えるようにいろいろなパラメータを追加や判断したことが原因である。新規追加部は DFD 用のテーブルに入出力データ情報を書込むものであった。再利用したモジュールのソースコードの再利用率は0.87であった。

FSD/DFD/PAD の設計知識を再利用する知識ベースを開発することを一本化した。大部分が小さいユニットになっており，これは各種の図形の統一表記による図形シンボル変換部の共通化もあるが，データ構造の標準化と機能を単機能化し，小さいユニットに分割して必要なら共用したので，システム全体では各ユニットの再利用率が高まり，全体の規模が小さく，より作りやすくなった。設計は小さなユニット毎に行うから，誤りが少なくなり，また，既にできているものを再利用することはバグの除去にかかる労力が減るので，作業効率の上昇につながった。信頼性も向上でき，再利用することによる開発期間の短縮/品質の保証ができた。

5.2　省力効果の評価

ソフトウェアシステムは，母体の開発以後に機能の拡張や変更（所謂保守）を重ね，長年月の間には母体開発コストよりも以後の変更コストの方が大きくなる。本方式はかような環境に適合するツールである。長期間にわたる正確な変更履歴の実例が得られなかったので，ここで，原理研究期の変更履歴を基に設計ルールを再利用する自動化部分について定量評価した。

図 6(a)は，横軸に上記の設計経験の累計数，縦軸はその時点までに使用した設計ルールの累計数を示す。初めにグラフは急激に増えるが，次第に増加が緩やかになる傾向を示し，設計

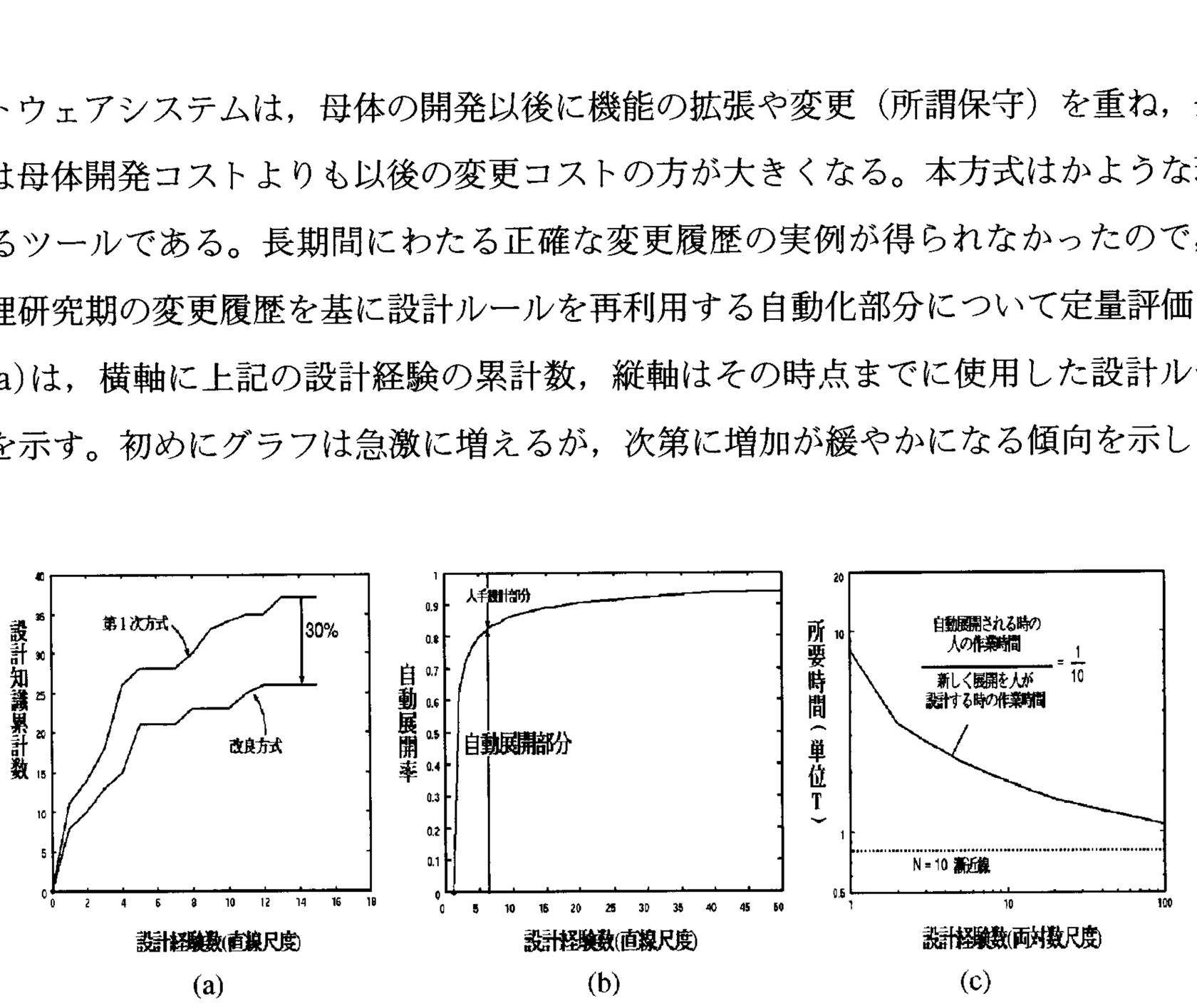

図 6　設計ルールを再利用する自動設計の特性

を度重ねる毎に新しい知識の追加の割合は低下する。両軸を対数尺度にとると，傾向線は直線になる。Industrial Engineering（IE）は，この傾向を対数習熟効果[8]と呼ぶ。図6(b)は縦軸に自動展開される率を示し，初めに急激に立上り80％を越すが中々100％に近づかず，浅知識の限界が見える。図6(c)は控え目に見たN＝10の場合の結果を示す。はじめに急激に下がり，経験10回目では，工数は1/4に低下する。設計には多様な作業方式が使われる。本方式のように早くから使われるものは習熟による効率化が早くに表れ，以後はより高度な作業方式が順次使われ始め，使われはじめた領域で習熟による効率化を示す。これら各種の習熟を総合した結果として，設計全体では広い範囲で習熟特性を示す。これ以上に経験数を大きくしてもなかなか上がらない。また，図5に示したようにほかの自動設計エンジン（設計ルールの自動生成部やプログラムの自動生成部）を使うことにより，自動化率を更に高くすることもできた[1]。

そこで，「この方式は早くに習熟効果が表れ，自動化率もかなり高く，大きく省力化できる」事が特長で，深追いしてもそれ以上の改善は少ない。エキスパートは人の思考の経済化のZip原則[14]に従って同一問題を解く多数の解法を備え，簡単な方法から始めて次第に高度な解法へと細やかに高度化させる。

6. ま　と　め

この論文は，ソフトウェア設計を対象として，自動設計システムにおけるエキスパートシステムの系統的な構築方法について述べた。この方式は，高い成熟度のソフトウェア開発組織を対象エキスパートとし，エキスパートの知識体系の基本モデルとして階層的な工程をとり，階層展開の概念の親子関係をそのプロセスの単位的な知識とする。設計図面から設計知識を系統的に獲得し，ボトムアップにそれを系統的にエキスパートシステムを再構築し，自動設計を行わせた。オープンエンドな系統的構成方式を説明し，更に各エキスパートユニットの構築にはソフトウェア工学的な系統的方法による事を述べた。設計情報と設計知識や各機能の分割などを標準化することより上流レベルのシステム設計からデータフロー図，構造化チャートとコーディングまでの自動化システムの開発ができ，かつ容易化することができた。

また，CASEツールとエキスパートシステムと組み合わせて統合知的CASEシステムの概要を述べ，今回の構築結果として作業方法の標準化と規模減少の評価や，自動設計システムとして基本的な詳細化の特性と知識の習熟効果を報告した。その省力効果を定量評価して，少数回数でも省力効果が得られ目的達成の見込みを得た。従来の自動設計方式のように，設計知識の事前準備を要することがなく，対象を問わない汎用性を持ち，しかも導入や使用が容易である。

謝　　辞

本研究は埼玉大学でソフトウェアクリエーションプロジェクトとして行われた。システム試作に貢献された堤永保氏はじめ当時日立中部ソフトウェア（現日立システムアンドサービス）の関係各位，カルガリー大学助教授 B. H. Far 博士，研究室の諸君のプロジェクトへの貢献に深く感謝します。また，日立製作所コンピュータ事業本部のご援助を頂いたことをここに記して謝意を表します。

参 考 文 献

[1] Abolhassani, H., Chen, H, and Koono, Z., "Software Creation: Cliche as Intermediate Knowledge in Software Design", the Journal of IEICE on Information and Systems, Vol. 85-D, No. 1, pp. 221-232, 2002.

[2] 陳　　慧，Far, B. H.，河野善彌："ソフトウェア自動設計における系統的なエキスパートシステムの構築，—設計工程からの設計知識の獲得と再現—"，人工知能学会誌，Vol. 12, No 4, pp. 616-626, 1997.

[3] Chen, H., Tsutsumi, N., Takano, H., and Koono, Z., "Software Creation: An Intelligent CASE Tool Featuring Automatic Design for Structured Programming", The Journal of Institute of Electronics, Information and Communication Engineers, Vol. E81-D, No. 12, 1439-1449. 1998.

[4] Chen, H, Takano, H., Abolhassani, H., Koono, Z., "Software Creation: an Integrated Environment for Automatic Software", Proceedings of the Fifth World Conference on Design and Process, Texas, U.S.A, pp. 27 (in conference CD-ROM), 2000.

[5] DeMarco, T.: "Structured Analysis and System Specification," Yourdon, 1979.

[6] Ford, K. M. and Bradshaw, J. M. eds., "Knowledge acquisition as modeling part 1", International Journal of Intelligent Systems, Vol. 8, No. 1, 1993.

[7] 二村良彦，"プログラム技法：PADによる構造化プログラミング"，オーム社，1984.

[8] Hancock, W. M., Bayer, F. H., "Ch2. Learning curve, Handbook of Industrial Engineering", John Wiley & Sons, 1982.

[9] Jackson M A. " Principles of Program Design.", Mass: Academic Press, 1975.

[10] Koono, Z., Far, B. H., Sugimoto, T., Ohmori, M. and Chen, H. "A Systematic Approach for Design Knowledge Acquisition from Documents", In Proceedings of 3rd Japanese Knowledge Acquisition for Knowledge-Based System Workshop, 253-265, 1994.

[11] Martin, J., "Information Engineering",. Prentice-Hall, 1989.

[12] Mayers G J., "Composite/Structured Design.", New York: Van Nostrand Reinhold, 1978.

[13] "Visio Manual", Visio Corporation, 1997.

[14] Zipf G. K., "Human Behavior and the Principle of Least Effort", Hafner Publishing, 1972.